UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/855,433	05/14/2001	David Ungar	83000.1113/P4157	9020

24209 7590 04/08/2004

GUNNISON MCKAY & HODGSON, LLP
1900 GARDEN ROAD
SUITE 220
MONTEREY, CA 93940

EXAMINER	
RAMPURIA, SATISH	

ART UNIT	PAPER NUMBER
2124	6

DATE MAILED: 04/08/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/855,433	UNGAR, DAVID	
	Examiner	Art Unit	
	Satish S. Rampuria	2124	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *03* MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *14 May 2001*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-6* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-6* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *09/25/2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *02/12/02, 06/05/02*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the application filed on 05/14/2001.

2. Claims 1-6 are pending.

Information Disclosure Statement

3. An initialed and dated copy of Applicant's IDS form 1449, Paper No. 4 and 5, is attached to the instant Office action.

Drawings

4. The drawing No. 6, filed on 09/25/2001 have been accepted by the examiner.

5. The drawing No. 1 is objected to because it is not clearly readable/viewable. A proposed drawing correction or corrected drawings are required in reply to the Office action to avoid abandonment of the application. The objection to the drawings will not be held in abeyance.

Specification

6. The abstract of the disclosure is objected to because it contains more than 150 words. Correction is required. See MPEP § 608.01(b).

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. Claims 1, 3, and 6 are rejected under 35 U.S.C. 103(a) as being unpatentable over Adi-Tabatabai, hereinafter called Adi-Tabatabai, US Patent No. 6,093,216 in view of Sumi et al., hereinafter called Sumi, US Patent No. 5,881,288.

Per claim 1, 3:

Adi-Tabatabai discloses:

- ***A method in a computer system for identifying at runtime a resource allocation error*** (col. 2, lines 34-35 "a method for run-time tracking of object references in computer code") ***generated by a virtual machine compiler*** (col. 4, lines 34-35 "a Java Virtual Machine that compiles the code before execution")

- ***obtaining a resource allocation indicator at runtime*** (col. 4, lines 54-55 "The memory space 400 in FIG. 4a comprises of a run-time stack 410, registers 450, static variables (462, 468, 472) and objects (460, 464, 466, 470) on the heap")

- ***testing at least one bit of said resource allocation indicator*** (col. 6, lines 23-24 "compiler checks to see if a variable will ever be assigned a reference value"), ***wherein said at least one bit corresponds to an allocated resource*** (col. 6, lines 8-10 "corresponding variable does not contain a reference type, then the bit is cleared")

- ***setting said at least one corresponding bit, when said at least one corresponding bit is unset, to indicate allocation of said corresponding resource*** (col. 6, lines 26-28 "JIT compiler... inserts... instruction... will set... variable's corresponding bit in the bit vector if the variable... store... reference type.")

- unsetting said at least one corresponding bit when said corresponding resource is deallocated (col.6, lines 29-32 "JIT compiler... insert... variable... instruction... clears... corresponding bit in the bit vector")

Adi-Tabatabai does not explicitly disclose halting exectuion.

However, Sumi discloses in an analogous computer system with halting exectuion (col. 19, line 39 "Execution halted after execution of Line YYY").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method for halting execution as taught by Sumi into the method for run time tracking of objects as taught by Adi-Tabatabai. The modification would be obvious because of one of ordinary skill in the art would be motivated to halt the program execution in debugging and optimization of program as suggested by Sumi (col. 1, lines 5-15).

Per claim 6:

- A computer system for processing register allocation in a computer comprising (col. 5, lines 52-53 "FIG. 5a shows some of the steps taken by the JIT compiler during compilation of computer code")

- a memory unit configured to store at least one allocation indicator (col. 5, line 55 "compiler allocates memory space")

- a plurality of registers (col. 5, lines 58-59 "physical machine registers")

- designating an address in said memory unit to represent an allocation indicator (col. 5, lines 59-60 "objects are assigned and become references")

- loading said allocation indicator into one of said plurality of registers (col. 6, lines 6-7 "a bit is set when the corresponding variable contains a reference type")

- testing a bit of said allocation indicator (col. 6, lines 23-24 "compiler checks to see if a variable will ever be assigned a reference value***"), to determine if said one of said plurality of registers is available for allocation*** (col. 6, lines 8-10 "corresponding variable does not contain a reference type, then the bit is cleared")

- setting said bit of said allocation indicator when it is unset to indicate that said one of said registers is unavailable for allocation (col. 6, lines 26-28 "JIT compiler... inserts... instruction... will set... variable's corresponding bit in the bit vector if the variable... store... reference type")

- clearing said at least one bit of said allocation indicator when said register is no longer used (col.6, lines 29-32 "JIT compiler... insert... variable... instruction... clears... corresponding bit in the bit vector" and col. 2, lines 10-12 "The space used by objects that will no longer be accessed ("dead objects") is freed by the garbage collector for future use").

Adi-Tabatabai does not explicitly disclose failing/halting execution.

However, Sumi discloses in an analogous computer system with failing/halting execution (col. 19, line 39 "Execution halted after execution of Line YYY").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method for halting execution as taught by Sumi into the method for run time tracking of objects as taught by Adi-Tabatabai. The modification would

be obvious because of one of ordinary skill in the art would be motivated to halt the program execution in debugging and optimization of program as suggested by Sumi (col. 1, lines 5-15).

9. Claim 2 is rejected under 35 U.S.C. 103(a) as being unpatentable over Adi-Tabatabai in view of Beadle et al., hereinafter called Beadle, US Patent No. 6,321,377.

Per claim 2:

The rejection of claim 1 is incorporated, and further, Adi-Tabatabai does not explicitly disclose generating an error condition when said at least one bit is set.

However, Beadle disclose in an analogous computer system execeptions generated during an execution (Abstract, "handling exceptions generated during an execution of instructions... exception is detected.... exception results from the execution of the instructions, wherein the exception has a location within the instructions"). A bit must be set if the exception is generated.

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of handling exceptions generated as taught by Beadle into the combination system of method run time tracking of objects and halting execution as taught by Adi-Tabatabai and Sumi. The modification would be obvious because of one of ordinary skill in the art would be motivated to generate error condition to efficiently debug the program as suggested by Beadle (col. 2, lines 17-32).

10. Claims 4 and 5 are rejected under 35 U.S.C. 103(a) as being unpatentable over Sumi in view of Adi-Tabatabai.

Per claim 4:

Sumi discloses:

- A method in a computer system of generating code to track at runtime the allocation of registers used in a virtual machine comprising (col. 1, lines 5-12 "a program development system... debugging apparatus... during software development... check... execution code generated... a program conversion apparatus... discover... errors present in such code... a debugging information generation apparatus for generating debugging information")

- generating code to allocate in memory (col. 1, line 63 "a generated code storage unit") a register mapping indicator (col. 1, lines 48-50 "allocation information that shows the correspondence between the variables in the source code and the registers and memory addresses used by the execution code")

- generating code to determine if a register is available for allocation comprising (col. 2, lines 27-28 "determines which resource (register) has been allocated the indicated variable")

- generating code to fail if said at least one bit of said register mapping indicator is set and allocation of said register is required (col. 2, lines 56-59 "Once it is established that Line 10 corresponds to address "0.times.128", the code execution unit 402 is ordered to set a breakpoint at address "0.times.128" and the line display unit 405 is ordered to display Line 10 with an arrow")

Sumi does not explicitly disclose variable mapping, test a bit before loading, and unsetting or clearing the bit.

However, Adi-Tabatabai discloses in an analogous computer system variable mapping, compiler checks the if the variable/register is assigned a value and clears the bit (and col. 7, lines 37-40 "the garbage collector looks at the variable mapping from the compiler to determine which variable the particular bit is assigned and where the variable is located as in step 585 and col. 6, lines 23-24 "compiler checks to see if a variable will ever be assigned a reference value" and (col. 6, lines 26-28 "JIT compiler... inserts... instruction... will set... variable's corresponding bit in the bit vector if the variable... store... reference type").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of variable mapping and testing bit and unsetting the bit as taught by Adi-Tabatabai into the method of optimizing the debugger as taught by Sumi. The modification would be obvious because of one of ordinary skill in the art would be motivated to variable mapping, test a bit before loading a value and clearing the bit to optimize the code as suggested by Adi-Tabatabai (col. 2, lines 13-31).

Per claim 5:

Sumi discloses:

- A method in a computer system for tracking temporary register allocated by a virtual machine comprising (col. 1, lines 5-12 "a program development system... debugging apparatus... during software development... check... execution code generated... a program conversion apparatus... discover... errors present in such code... a debugging information generation apparatus for generating debugging information" and col. 4, lines 5-6 "program conversion apparatus generates temporary variables")

- ***allocating a temporary register mapping indicator*** (col. 1, lines 48-50 "allocation information that shows the correspondence between the variables in the source code and the registers and memory addresses used by the execution code" and col. 4, lines 5-6 "program conversion apparatus generates temporary variables")

- ***determining if a register is available for allocation by executing the steps comprising*** (col. 2, lines 27-28 "determines which resource (register) has been allocated the indicated variable" and col. 4, lines 5-6 "program conversion apparatus generates temporary variables)

- ***failing if said at least one bit of said temporary register mapping indicators is set*** (col. 2, lines 56-59 "Once it is established that Line 10 corresponds to address "0.times.128", the code execution unit 402 is ordered to set a breakpoint at address "0.times.128" and the line display unit 405 is ordered to display Line 10 with an arrow" and col. 4, lines 5-6 "program conversion apparatus generates temporary variables)

Sumi does not explicitly disclose test a bit before loading and unsetting or clearing the bit.

However, Adi-Tabatabai discloses in an analogous computer system variable mapping, compiler checks the if the variable/register is assigned a value and clears the bit (and col. 7, lines 37-40 "the garbage collector looks at the variable mapping from the compiler to determine which variable the particular bit is assigned and where the variable is located as in step 585 and col. 6, lines 23-24 "compiler checks to see if a variable will ever be assigned a reference value" and (col. 6, lines 26-28 "JIT compiler... inserts... instruction... will set... variable's corresponding bit in the bit vector if the variable... store... reference type").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of variable mapping and testing bit and unsetting the bit as taught by Adi-Tabatabai into the method of optimizing the debugger as taught by Sumi. The modification would be obvious because of one of ordinary skill in the art would be motivated to variable mapping, test a bit before loading a value and clearing the bit to optimize the code as suggested by Adi-Tabatabai (col. 2, lines 13-31).

Conclusion

11. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

The following patent is cited to further show the state of the art with respect to optimizing the debugging of compiler.

US Patent No. 5,659,754 to Grove et al.

US Patent No. 6,263,489 to Olsen et al.

Publication No. JP- 63-045637 to Ryoji

Any inquiry concerning this communication or earlier communication from the examiner should be directed to Satish Rampuria whose telephone number is 703-305-8891.

The examiner can normally be reached on Monday-Friday from 8:30 A. M. to 5:00 P.M. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor Kakali Chaki can be reached at 703-305-9662. The fax number for this group is 703-872-9306.

An inquiry of general nature or relating to the status of this application or proceeding should be directed to the group receptionist whose telephone number is 703-305-3900.

Satish S. Rampuria

Patent Examiner

Art Unit 2124

04/05/04

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100